IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Applicant:	Mortazavi et al.	
App. No.:	09/865,978	Con. No.: 6345
Filed:	May 25, 2001	Art Unit: 2152
Title:	METHOD AND APPARATUS FOR ASYNCHRONOUS COMPONENT INVOCATION	Examiner: Lesniewski, Victor D.

RESPONSE TO THE THIRD NOTICE OF NON-COMPLIANT APPEAL BRIEF

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

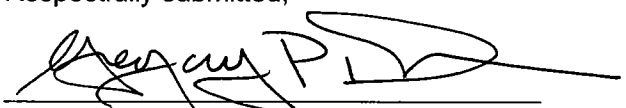
Applicant (hereafter "Appellant") hereby submits the Appendix of Claims in response to the to the Third Notice of Non-Compliant Appeal Brief mailed March 6, 2009 in the above-captioned case. The enclosed Appendix is corrected to be consistent with the claims filed in the last Amendment dated September 15, 2006. We note that this information was previously provided; however, the claims presented in the Appendix were slightly inconsistent with the last Amendment filed.

Appellant respectfully requests consideration of this appeal by the Board of Patent Appeals and Interferences (hereafter the "Board").

Appellant believes that no fees or petitions are required are required at this time. However, if any such petitions or fees are necessary, please consider this a request therefor and authorization to charge Deposit Account No. 04-1415 accordingly.

Dated: March 19, 2009

Respectfully submitted,



Gregory P. Durbin, Registration No. 42,503
Attorney for Appellant
USPTO Customer No. 66083

DORSEY & WHITNEY LLP
Republic Plaza Building, Suite 4700
370 Seventeenth Street
Denver, Colorado 80202-5647
Phone: (303) 629-3400
Fax: (303) 629-3450

IX. APPENDIX OF CLAIMS

1. A computer-implemented method for a first component to invoke a second component asynchronously in an object-oriented computing environment, the computer-implemented method comprising:

receiving at an asynchronous proxy an asynchronous request from a first object-oriented component residing at a first server to invoke a second object-oriented component residing at a second server wherein the request has a void return type and is not associated with application-specific exceptions;

setting an exception listener on the asynchronous proxy and a scope of the second component, the exception listener being registered for the second component;

storing the request and the scope in a queue on the asynchronous proxy; and

providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener object-oriented component for handling exceptions associated with the invocation of the second component, wherein the exception listener is registered on an asynchronous proxy, is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

4. The computer-implemented method of claim 1, wherein the first and second components reside in environments allowing components to directly invoke other components.

5. The computer-implemented method of claim 1, wherein the first and second components are Enterprise Java Bean components.

6. The computer-implemented method of claim 5, wherein the first and second components are associated with a container.

7. The computer-implemented method of claim 6, further comprising placing the request from the first component in a queue.

8. The computer-implemented method of claim 7, wherein a worker thread dequeues the received request after receiving a transaction commit signal from the container.

9. The computer-implemented method of claim 8, wherein the exception listener receives the exception and the scope of the exception.

10. A computer-implemented method for a first object-oriented component to invoke a second object-oriented component asynchronously in an object-oriented environment, the computer-implemented method comprising:

transmitting an asynchronous request from the first object-oriented component residing at a first server to invoke the second object-oriented component residing at a second server, the first and second object-oriented components operating in environments allowing direct invocation of the second component by the first component;

receiving the asynchronous request wherein the request has a void return type and is not associated with application-specific exceptions; and

registering an exception listener object-oriented component on an asynchronous proxy and a scope of the second object-oriented components, the exception listener being registered for the second component and, the asynchronous proxy being associated with the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

11. The computer-implemented method of claim 10, wherein the asynchronous proxy has the same type as the second component.

13. The computer-implemented method of claim 10, wherein the first and second components are associated with separate servers.

14. The computer-implemented method of claim 10, wherein the first and second components are Enterprise Java Bean components.

15. The computer-implemented method of claim 14, wherein the first and second components are associated with a container.

16. A computer program product comprising computer code for a first component to invoke a second component asynchronously, the computer program product comprising:

computer code for receiving at an asynchronous proxy an asynchronous request from a first object-oriented component residing at a first server to invoke a second object-oriented component residing at a second server, wherein the request has a void return type and is not associated with application-specific exceptions;

computer code for setting an exception listener on the asynchronous proxy and a scope of the second component, the exception listener being registered for the second component;

computer code for storing the request and the scope in a queue on the asynchronous proxy;

computer code for providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener object-oriented component for handling exceptions associated with the invocation of the second component, wherein the exception listener is registered on an asynchronous proxy, is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components; and

a computer-readable medium for storing the computer codes.

19. The computer program product of claim 16, wherein the first and second components reside in environments allowing components to directly invoke other components, wherein the first and second components are associated with separate servers.

20. The computer program product of claim 16, wherein the first and second components are Enterprise Java Bean components.

21. The computer program product of claim 20, wherein the first and second components are associated with a container.

22. The computer program product of claim 21, further comprising placing the request from the first component is placed in a queue.

23. The computer program product of claim 22, wherein the worker thread dequeues the received request after receiving a transaction commit signal from the container.

24. The computer program product of claim 23, wherein the exception listener receives the exception and the scope of the exception.

25. A computer program product for a first object-oriented component to invoke a second object-oriented component asynchronously in an object-oriented environment, the computer program product comprising:

computer code for transmitting an asynchronous request from a first object-oriented component residing at the first server to invoke the second object-oriented component residing at a second server, the first and second object-oriented components operating in environments allowing direct invocation of the second component by the first component;

computer code for receiving the asynchronous request wherein the request has a void return type and is not associated with application-specific exceptions;

computer code for registering an exception listener object-oriented component on an asynchronous proxy, and for setting a scope associated with the second object-oriented components, the exception listener being registered for the second component and the asynchronous proxy associated with the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components; and

a computer-readable medium for storing the computer codes.

26. The computer program product of claim 25, wherein the asynchronous proxy has the same type as the second component.

28. The computer program product of claim 25, wherein the first and second components are associated with separate servers.

29. The computer program product of claim 25, wherein the first and second components are Enterprise Java Bean components.

30. The computer program product of claim 29, wherein the first and second components are associated with a container.

31. A computer program product for an enterprise environment associated with a computing system, the computer program product comprising:

an asynchronous proxy for receiving a request from a first object-oriented component residing at a first server, the request intending to invoke a second object-oriented component residing at a second server wherein the request has a void return type and is not associated with application-specific exceptions; and

an exception listener object-oriented component coupled to the asynchronous proxy and registered for the second object-oriented component, wherein the exception listener uses a scope corresponding to the request to handle exceptions associated with the invocation of the second object-oriented component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

35. The computer program product of claim 31, wherein the first and second components are Enterprise Java Bean components.

36. The computer program product of claim 35, wherein the first and second components are associated with a container.

37. The computer program product of claim 31, wherein the worker thread invokes the second components after receiving a transaction commit signal from the container.

38. A computer system operating an enterprise environment, the computer system comprising:

a processor coupled to memory; and

an interface coupled to the processor, the interface configured to transmit a request from a first object-oriented component residing at the first server to invoke a second object-oriented component residing at a second server, wherein the request has a void return type and is not associated with application-specific exceptions the first and second object-oriented components operating in environments allowing direct invocation of the second component by the first component, wherein the interface also transmits information to register an exception listener on an asynchronous proxy, the asynchronous proxy associated with the second component and the exception listener being registered for the second component and, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

42. The computer system of claim 38, wherein the first and second components are Enterprise Java Bean components.

43. An apparatus for a first component to invoke a second component asynchronously in an object-oriented computing environment, the apparatus comprising:

means for receiving at an asynchronous proxy an asynchronous request from a first object-oriented component residing at a first server to invoke a second object-oriented component residing at a second server, wherein the request has a void return type and is not associated with application-specific exceptions;

means for setting an exception listener on the asynchronous proxy and a scope of the second component, the exception listener being registered for the second component;

means for storing the request and the scope in a queue on the asynchronous proxy; and

means for providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener object-oriented component for handling exceptions associated with the invocation of the second component, the exception listener object-oriented component registered on an asynchronous proxy, wherein the request is associated with no application specific exceptions.